Mesh Partitioning for Distributed Systems

Jian Chen Valerie Taylor
Northwestern University
Department of Electrical and Computer Engineering
Evanston, IL 60208
{jchen,taylor}@ece.nwu.edu

Abstract

Distributed systems, which consist of a collection of high performance systems interconnected via high performance networks (e.g. ATM), are becoming feasible platforms for execution of large-scale, complex problems. In this paper, we address various issues related to mesh partitioning for distributed systems. These issues include the metric used to compare different partitions, efficiency of the application executing on a distributed system, the number of cut sets, and the advantage of exploiting heterogeneity in network performance. We present a tool called PART, for automatic mesh partitioning for distributed systems. The novel feature of PART is that it considers heterogeneities in the application and the distributed system. The heterogeneities in the distributed system include processor and network performance; the heterogeneities in the application include computational complexities. Preliminary results are presented for partitioning regular and irregular finite element meshes for the WHAMS2D application executing on a distributed system consisting of two IBM SPs. The results from the regular problems indicate a $33-46\%$ *increase in efficiency when processor performance is considered as compared to the conventional even partitioning; the results also indicate an additional* $5-16\%$ *increase in efficiency when network performance is considered. The result from the irregular problem indicate a* 21% *increase in efficiency when processor and network performance are considered as compared to even partitioning.*

1 Introduction

Mesh partitioning for homogeneous systems has been studied extensively [1, 2, 5, 15, 17, 18, 24]; however, mesh partitioning for distributed systems is a relatively new area of research. To ensure efficient execution on a distributed system, the heterogeneities in the processor and network performance must be taken into consideration in the partitioning process; equal size subdomains and small cut set size, which results from conventional mesh partitioning, are no longer the primary goals. For distributed systems, the primary goals involve exploiting the system heterogeneities. In this paper, we discuss the major issues in mesh partitioning for distributed systems. These issues include the comparison metric, efficiency, cut sets, and exploiting heterogeneity in network performance. We also present PART, a mesh partitioning tool for distributed systems along with some preliminary results.

Distributed computing has been unequivocally regarded as the future of high performance computing. Nationwide high speed networks such as vBNS [13] are becoming widely available to interconnect high speed computers at different sites. Work is in progress to make available a National Technology Grid which will entail a scalable distributed computational fabric connecting supercomputers, virtual environments, scientific instruments, and large datasets [21]. Projects such as Globus [6] and Legion [10] are developing software infrastructure for computations that integrate distributed computational and informational resources.

Mesh partitioning is used for efficient parallel execution of mesh-based applications, which use techniques such as finite element and finite difference. These techniques are widely used in many disciplines such as biomedical engineering, structural mechanics, and fluid dynamics. These applications are distinguished by the use of a meshing procedure for discretizing the problem domain. Implementing a mesh-based application on a parallel or distributed system involves partitioning the mesh into subdomains that are assigned to individual processors in the parallel or distributed system. For a distributed system, a desirable partitioning method should take into consideration various heterogeneous features of the systems. PART takes advantage of the following heterogeneous system features: (1) processor speed; (2) number of processors; (3) local network performance; (4) wide area network performance. Further, different mesh-based applications may have different com-

putational complexities, different communication patterns, and different element types, which also must be taken into consideration when partitioning.

In [22], we presented a manual method of mesh partitioning for distributed systems; in [23], we presented an automatic mesh partitioning tool, PART. In this paper, however, we focus on the theoretical framework of mesh partitioning for mesh-based applications on a distributed system. In particular, we identify a good metric to be used to compare different partitioning results, present a measure of efficiency for a distributed system, discuss restrictions on the cut set for remote communication, and identify the conditions for which it is advantageous to consider heterogeneities in network performance in addition to processor performance.

The metric used with PART to identify good efficiency is estimated execution time. PART was used to partition regular and irregular meshes for a 2-D explicit finite element application, WHAMS2D, for execution on two IBM SPs located at geographically different sites: Argonne National Laboratory and Cornell Theory Center. For the regular meshes with 10368 elements, the initial results indicate an efficiency of $0.61 - 0.67$ when only processor performance is considered, as compared to $0.46 - 0.47$ for conventional methods. When heterogeneity in both processor and network performance were considered, the efficiency was $0.68 - 0.71$, which is a substantial increase over the conventional methods of equal size subdomains and small cut set size. For the irregular mesh Barth4 with 11451 elements, the initial results indicate an efficiency of 0.85 when both processor and network performance were considered, as compared to an efficiency of 0.70 for conventional methods.

The remainder of the paper is organized as follows: Section 2 provides background. Section 3 discusses issues. Section 4 describes PART in detail. Section 5 is experimental results. Section 6 gives previous work and finally conclusion.

2 Background

2.1 Mesh Based Applications

Finite element method has been a fundamental numerical analysis technique to solve partial differential equations in the engineering community for the past three decades. The method includes three basic procedures. The problem is first formulated in variational or weighted residual form. In the second step, the problem domain is discretized into complex shapes called elements. The last major step is to solve the resulting system of equations. The procedure of discretizing the problem domain is called meshing. Applications that involves a meshing procedure are referred to as mesh-based applications. In addition to finite element method, finite difference method and finite volume method also utilize a meshing procedure. The meshing procedure is an important first step in the analysis, especially when complex geometries are considered. The quality of the numerical results is strongly linked to the quality of the corresponding mesh; finer meshes result in more accurate results.

Mesh based applications are naturally suited for parallel or distributed systems. Implementing the finite element method in parallel involves partitioning the global domain of elements into P connected subdomains that are distributed among P processors; each processor executes the numerical technique on its assigned subdomain. The communication among processors is dictated by the types of integration and solver methods. Explicit integration finite element problems do not require the use of a solver since a diagonal matrix (the lumped matrix) is used. Therefore, communication only occurs among neighboring processors that have common data and is relatively simple. For implicit integration problems, however, communication is determined by the type of solver used in the application. The application used in this paper is an explicit, nonlinear finite code, called WHAMS2D [3], which is used to analyze elastic plastic materials. While we focus on the WHAMS2D, the concepts can be generalized to other mesh-based applications.

2.2 Distributed System

Distributed computing consists of a platform with a network of resources. These resources maybe clusters of workstations, cluster of personal computers, or parallel machines. Further, the resources maybe located at one site or distributed among different sites. Figure 1 shows an example of a distributed system.

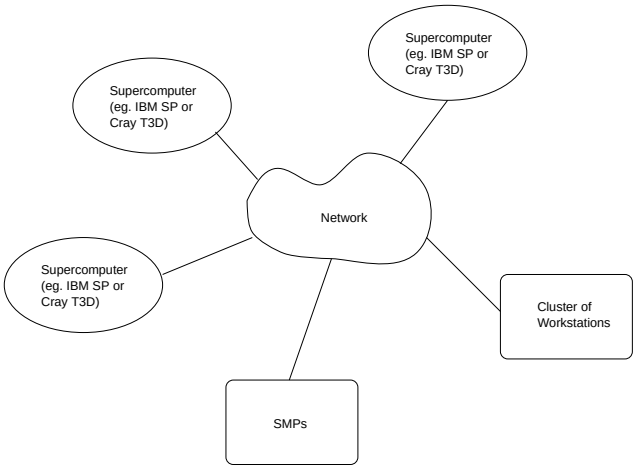

Figure 1. A distributed system.

Distributed systems provide an economical alternative to costly massively parallel computers. Researchers are no longer limited to the computing resources at individual sites. The distributed computing environment also provides researchers opportunities to collaborate and share ideas through the use of collaboration technologies.

In a distributed system, we define *"group"* as a set of processors that share one interconnection network and have the same performance. A group can be an SMP, a parallel computer, or a cluster of workstations. Communication occurs both within a group and between groups. We refer to communication within a group as *"intra"* group communication, and communication between processors in different groups as *"inter"* group communication.

2.3 Problem Formulation

Mesh partitioning for homogeneous systems can be viewed as a graph partitioning problem; the goal of the graph partitioning problem is to find a small vertex separator and *equal* sized subsets. Mesh partitioning for distributed systems, however, is a variation of the graph partitioning problem; the goal differs in that equal sized subsets may not be desirable. The partitioning problem for distributed systems can be stated as follows:

Given a graph $G = (V, E)$, with $|V| = n$ for positive integer n, and weights $w(v) \in Z^+$ for each $v \in V$ and $l(e) \in Z^+$ for each $e \in E$, partition V into $k \leq n$ subsets, $V_1, V_2, ..., V_k$, such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $\bigcup_{1 \leq i \leq k} V_i = V$, and $\sum_{i=1}^{k} (f_i)^2 \leq J$, where J is a positive integer, and f_i is a function of $w(v)$ with $v \in V_i$ and $l(e)$ with e being the edge that has one end point in V_i and the other end point in V_j, $j \neq i$.

In this paper, the cost function f_i is an estimate of the execution time of partition i of a given application on a distributed system. Minimizing the sum of squares of f_i, $1 \leq i \leq k$, minimizes the variance of the execution time of all the processors, thereby resulting in balanced execution among the processors. This cost function is discussed further in [23].

The partitioning problem for distributed system is NP-complete; the proof is given in the Appendix. Therefore, we focus on heuristics to solve this problem.

2.4 Nomenclature

Throughout this paper, the following nomenclature is used:

E_i	–	Estimated execution time on processor i.
E_{comp_i}	–	Estimated computational time on processor i.
E_{comm_i}	–	Estimated communication time on processor i.
F_i	–	Performance of processor i as measured by a computation kernel.
α_{Li}	–	Per message cost of the i-th intra group message.
α_{Ri}	–	Per message cost of the i-th inter group message.
β_{Li}	–	Per byte cost of the i-th intra group message.
β_{Ri}	–	Per byte cost of the i-th inter group message.
L_i	–	Size of the i-th intra group message.
R_i	–	Size of the i-th inter group message.
γ	–	Coefficient of computational complexity.
λ	–	Parameter used to equalize the contribution of the computation and communication to execution time
$N^i{}_{elem}$	–	Number of elements in partition i.
$N^i{}_{interface}$	–	Number of interface elements in partition i.
N_e	–	Number of elements.

3 Major Issues

In this section, we discuss the following major issues related to the mesh partitioning problem for distributed systems: comparison metric, efficiency, conditions for which network performance need to be considered, and number of cuts between groups.

3.1 Comparison Metric

The de facto metric for comparing the quality of different partitions has been minimum interface (or cut set) size assuming equal size partitions. Although there have been counter examples [5], this metric has been used extensively in comparing the quality of different partitions. It is obvious that equal subdomain size and minimum interface is not valid for comparing partitions for distributed systems.

One may consider an obvious metric for a distributed system to be unequal subdomains (proportional to processor performance) and small cut set size. The problem with this metric is that heterogeneity in network performance is not considered. Given the local and wide area networks are used in distributed systems, it is the case that there maybe a large difference between intra and inter group communication, especially in terms of latency. This difference must be considered to achieve efficient execution.

We argue that the use of an estimate of execution time of the application on the target heterogeneous system will always lead to a valid comparison of different partitions. The task of getting an accurate estimate of execution time for an application on a given distributed system maybe difficult. However, the estimate is used for relative comparison of different partition methods. Hence the focus is not on the accuracy of the estimate, rather it is the relative comparison that makes the estimate useful. It is important to make the estimate representative of the application and the system. The estimate should take into consideration the system heterogeneities such as processor performance, intra and inter group communication. It should also reflect the application computational complexity.

3.2 Efficiency

An important issue is how efficient is the execution of the application on a distributed system. For the case of homogeneous machines, efficiency is the ratio of the relative speedup to perfect speedup. The efficiency for the distributed system, however, is complicated by the fact that the system includes processors with different performance. Therefore, the efficiency for the distributed system is equal to the ratio of the relative speedup to the effective number of processors, V. This ratio is given as follows:

$$Efficiency = \frac{E(1)}{E \times V}$$

$E(1)$ is the sequential execution time on one processor; E is the execution time on the distributed system. The term V is equal to the summation of each processor's performance relative to the performance of the processor used for sequential execution. This term can be written as follows:

$$V = \sum_{i=1}^{P} \frac{F_i}{F_k}$$

where k is the processor used for sequential execution. For example, with two processors having processor performance $F_1 = 1$ and $F_2 = 2$ (in relative terms), the efficiency would be $Efficiency = \frac{E(1)}{E \times 3}$ if processor 1 is used for sequential execution; the efficiency is $Efficiency = \frac{E(1)}{E \times 1.5}$ if processor 2 is used for sequential execution.

3.3 Heterogeneous Networks

It is well-known that heterogeneity of processor performance must be considered with distributed systems. In this section, we identify conditions for which heterogeneity in network performance must be considered. This is achieved by considering a simple case, stripe partitioning, for which communication occurs with at most two neighboring processors. Assume there exists two groups having the same processor and local network performance; the groups are located at geographically distributed sites requiring a WAN for interconnection. Figure 2 illustrates one such case.

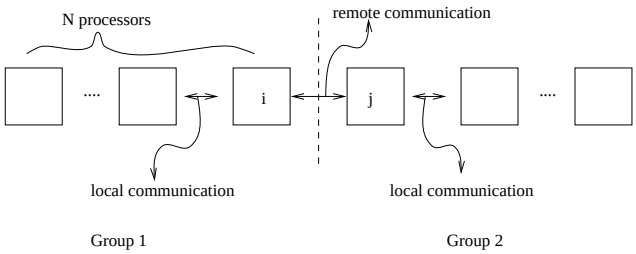

Figure 2. Communication Pattern for Stripe Partitioning.

Processor i (as well as processor j) requires local and remote communication. The difference between the two communication times is:

$$C_R - C_L = x\%E$$

where x is the percentage of the difference of C_R and C_L in the total execution time E. Assume that E represents the execution time taking into consideration only processor performance. Since it is assumed that all processors have the same performance, this entails an even partition of the mesh. This time can be written as:

$$E = C_L + C_R + E_{comp} = 2C_L + x\%E + E_{comp}.$$

Now consider the case of partitioning to take into consideration the heterogeneity in network performance. This is achieved by decreasing the load assigned to processor i and increasing the loads of the $N-1$ processors in group 1. The same applies to processor j in group 2. The amount of the load to be redistributed is $C_R - C_L$ or $x\%E$ and this amount is distributed to N processors. This is illustrated in Figure 4, which is discussed with the retrofit step of PART. The execution time is now:

$$E' = 2C_L + \frac{x}{N}\%E + E_{comp}.$$

The difference between E and E' is:

$$\Delta E = E - E' = x\%E - \frac{x}{N}\%E = \frac{N-1}{N}x\%E \simeq x\%E$$

Therefore, by taking the network performance into consideration when partitioning, the percentage reduction in execution time is approximately x. The percentage x is determined by (1) the percentage of communication in the application (y) and (2) the difference in the remote and local

communication (z). Both factors are determined by the application and the partitioning.

For the WHAMS2D application, y is 0.10, and z is 10 for vBNS (corresponding to an order of magnitude difference between the IBM SP Vulcan switch and vBNS). For this case, considering heterogeneity in network performance did not provide any significant increase in efficiency beyond that equal to variance in the results. However, with the Internet, having two orders of magnitude difference, considering network performance provided an additional $5 - 17\%$ increase in efficiency as compared to only considering processor performance. Hence, for this application, communication was a small fraction of the overall execution, but the large difference in network latencies resulted in performance improvements when considering heterogeneity in networks.

The WHAMS2D application uses explicit integration, which entails very simple communication. Implicit methods, however, entails more complex communication patterns. In this case it maybe advantageous to exploit heterogeneity in network performance even with vBNS.

3.4 Number of Cuts Between Groups

Once a cut has been made between groups, care must be taken when partitioning around this cut within a group. This is a major issue because of the large difference in latency between local and wide area networks. In particular, for the processor(s) that requires remote communication, care must be taken to reduce the number of remote processors with which this processor must communicate. For example, with the IBM SP, the latency of the Vulcan switch for MPI communication is 0.9 ms as compared to 10 ms for the vBNS network between Argonne National Laboratory and Cornell Theory Center (an order of magnitude difference). When this difference is large, it is advantageous to decrease the number of messages sent remotely, thereby decreasing latency; this may result in an increase in the number of messages sent within a group. This tradeoff must be considered when partitioning.

4 PART

PART considers heterogeneities in both the application and the system. In particular, PART takes into consideration that different mesh-based applications may have different computational complexities and the mesh may consist of different element types. For distributed systems, PART takes into consideration heterogeneities in processor and network performance.

Figure 3 shows a flow diagram of PART, which consists of an interface program and a partitioning program. The input to PART is a mesh that is transformed to a weighted communication graph. This communication graph is used by the partitioning program to produce the desired subdomains for execution on a distributed system. The partitioning program uses simulated annealing for the required retrofitting steps discussed below. While it is well known that simulated annealing is computationally intensive, it is used with the initial version of PART to provide the necessary local and global optimizations. We are currently investigating the use of parallel simulated annealing methods to significantly reduce execution time.

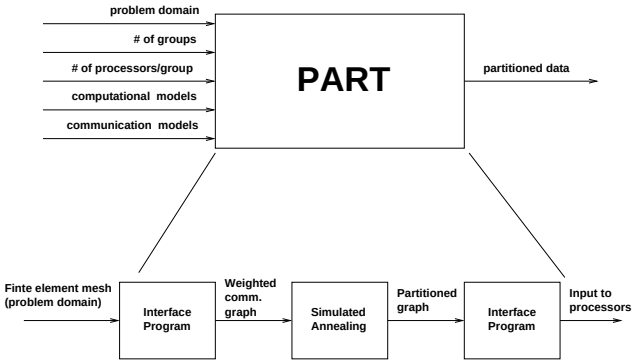

Figure 3. PART owchart.

4.1 Mesh Representation

The first component of PART converts an input mesh into a weighted communication graph (WCG). In the WCG, each vertex represents an element, and the weight on each vertex represents the number of nodes in the element. Each edge in the WCG indicates that the element shares nodes with a neighboring element. The weight on an edge represents the number of nodes the neighboring elements share. The WCG graph allows for ease of estimation of execution time since message size can be accurately represented for the case when the mesh consists of different element types.

4.2 Partition Method

The partitioning component of PART entails three steps:

1. The first step generates a coarse partitioning for the distributed systems. Each group gets a subdomain that is proportional to its number of processors, the performance of the processors, and the computational complexity of the application. Hence computational cost is balanced across all the groups.

2. In the second step, the subdomain that is assigned to each group from Step 1 is partitioned among its processors. Within each group, simulated annealing is used to balance the execution time. In this step, variance

in network performance is considered. Processors that perform inter group communication will have reduced computational load to compensate for the longer communication time.

The step is illustrated in Figure 4 for two supercomputers (or groups), SC1 and SC2, with four processors used for SC1 and two for SC2. The figure contains the load on the four processors of SCI. On the left side, the computational load is equal across the four processors since they have the same performance. For this case processor $P3$ has a large execution time because of the need to communicate remotely as well as locally. The right side displays the results after a retrofit step, in which part of the computational load has been removed from processor $P3$ (δ) and redistributed to the other three processors to equalize the execution time ($\delta/4$). Assuming the cut size remains unchanged (the communication time will not change) the execution time will be balanced after this shifting of computational load.

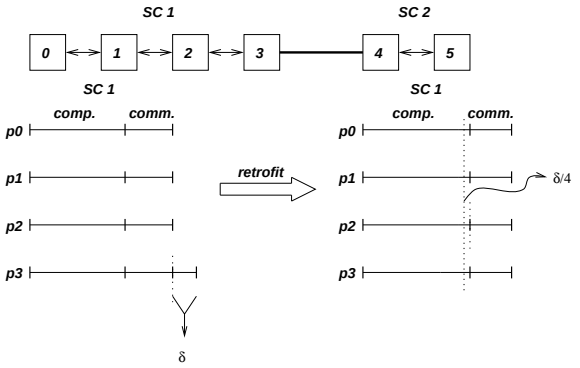

Figure 4. An illustration of the retrofit step for two supercomputers assuming only two nearest neighbor communication.

3. The third step takes into consideration differences in the local interconnect performance of various groups. Again, the goal is to minimize the variance of the execution time across all processors. In this step, elements on the boundaries of partitions are moved according to the execution time variance between neighboring processors. This step is only executed if there is a "large" difference in the performance of the different local interconnects. "Large" is defined by the users. The default value is 10%. For the case when a significant number of element is moved between the groups in Step 3, the second step is executed again to equalize the execution time in a group given the new computational load.

5 Experiments

PART was applied to an explicit, nonlinear finite code called WHAMS2D [3]. The code uses MPI built on top of Nexus for interprocessor communication within a supercomputer and between supercomputers. Nexus is a runtime system that allows for multiple protocols within an application [7]. The computational complexity of WHAMS2D is linear with the size of the problem.

The code was executed on the IBM SP machines located at Argonne National Laboratory and the Cornell Theory Center. These two machines are connected by the Internet. Macro benchmarks were used to determine the network and processor performance. The results of the network performance analysis are given in Table 1. Further, experiments were conducted to determine that the Cornell nodes were 1.6 times faster than the Argonne nodes.

Table 1. Values of α and β for the different networks.

Argonne SP Switch	$\alpha_1 = 0.0009$s	$\beta_1 = 0.0001$ s/KB
Cornell SP Switch	$\alpha_2 = 0.0006$s	$\beta_2 = 0.0001$ s/KB
Internet	$\alpha_4 = 0.1428$s	$\beta_4 = 0.0507$ s/KB

The problem mesh consists of 3 regular meshes and one irregular mesh, barth4 with 11451 elements. Barth4 is a unstructured 2-dimensional finite element meshes of airfoils. The execution time is given for 100 time steps corresponding to 0.005 seconds of application time. The recorded execution time represents over 100 runs, taking the data from the runs with standard deviation less than 3%.

The regular problems were executed on a machine configuration of 8 processors (4 at ANL IBM SP and 4 at CTC IBM SP). The irregular problem barth4 was executed on a machine configuration of 4 processors (2 at ANL IBM SP and 2 at CTC IBM SP). The number of processors was dictated by availability.

Table 2 presents the results for the regular problems. Column 1 is the mesh configuration. Column 2 is the execution time resulting from the conventional equal partitioning using spectral bisection in the Chaco package [11]. Column 3 is the result from the partitioning for which the heterogeneity in processor performance and computational complexity are considered. Column 4 is the execution time resulting from the partitioning for which the heterogeneity in network and processor performance are considered. The results in Table 2 shows that approximately $33 - 46\%$ increase in efficiency can be achieved by balancing the computational cost; another $5 - 16\%$ efficiency increase can

Table 2. Execution time using the Internet 8 processors: 4 at ANL, 4 at CTC

Case	Chaco	Proc. Perf.	Local Retrofit
9×1152 mesh	102.99 s	78.02 s	68.81 s
$efficiency$	0.46	0.61	0.71
18×576 mesh	101.16 s	78.87 s	72.25 s
$efficiency$	0.47	0.61	0.68
36×288 mesh	103.88 s	73.21 s	70.22 s
$efficiency$	0.46	0.67	0.70

be achieved by considering the variance in network performance. The small increase in efficiency for considering the network performance is attributed to the fact that communication is only 10% of the execution time.

The global optimization step, which is the last step of PART that balances execution time across all supercomputers, was not executed because the difference in the two switches is small (less than 10%). This is expected since the two supercomputer we used, the Argonne IBM SP and the Cornell IBM SP, both have interconnection networks that have very similar performance as indicated in Table 1.

Table 3. Execution time (in seconds) using the Internet on 4 processors: 2 at ANL, 2 at CTC.

Case	Chaco	Proc. Perf. & Local Retrofit
barth4	203.88 sec.	166.92 sec.
efficiency	0.70	0.85

Table 2 presents the results for the irregular problem barth4. Column 1 is the mesh configuration. Column 2 is the execution time resulting from the conventional even partitioning that uses spectral bisection. Column 3 is the execution time resulting from the partitioning taken from PART considering network and processor performance . The results in Table 2 shows that 21% increase in efficiency can be achieved by taking the system as well as application features into consideration when partitioning.

6 Previous Work

Significant work has been done with developing mesh and graph partitioning methods. The well-known algorithms include: Kernighan-Lin [15], Greedy [5], Recursive Coordinate Bisection (RCB) [2], Recursive Inertial Partitioning (RIP) [16], Recursive Graph Bisection (RGB) [9], Recursive Spectral Bisection (RSB) [18], and multilevel methods used with Greedy and RSB [12, 14]. Most of the aforementioned decomposition methods are implemented in one of the five automated tools: Chaco [11], TOP/DOMDEC [19], JOSTLE [25], METIS [14], and HARP [20]. The original goal of these tools is to develop a partition with equal size subdomains and small cut set size. Modifications can be made to the input of most of these tools to take into consideration processor performance. This modification entails calculating the percentage of the domain to be assigned to each processor prior to using the tools. This calculation, however, is not required with PART. Further, the retrofit needed to take into consideration difference in network performance, is not achievable with the given tools.

A partitioning advisory system was presented in [4] for network of workstations. The advisory system takes into consideration the variance in processor performance among the workstations. The problem, however, is that linear computational complexity is assumed for the application. This is not the case with implicit finite element problems, which are widely used. Further, only regular problems were considered. PART works with irregular as well as regular meshes.

7 Conclusion

In this paper, we addressed issues in mesh partitioning for distributed systems. These issues include the metric used to compare different partitions, efficiency of the application executing on the distributed system, the number of cut sets, and the advantages of exploiting heterogeneity in network performance. In particular, estimated execution time was suggested for the comparison metric; we also derived a formula for efficiency that takes into account heterogeneity in processor performance. In terms of heterogeneous networks, one must consider the difference in the network performance as well as the application communication requirements to determine if the networks must be considered. For the WHAMS2D application, it was advantageous to exploit the heterogeneous networks when there was a two orders of magnitude difference. Lastly, the number of cut sets between groups must be minimized when possible.

We present the PART tool, which incorporates the aforementioned issues into a tool for mesh partitioning. The novel feature of PART is that it considers heterogeneities in both the application and the distributed system. Preliminary results are presented for partitioning 3 regular meshes and an irregular finite element mesh for the WHAMS2D application executing on a distributed system consisting of two supercomputers. The results from the regular problems

indicate a $33-46\%$ increase in efficiency when processor performance is considered as compared to even partitioning; the results also indicate an additional $5-16\%$ increase in efficiency when network performance is considered. The result from the irregular problem indicate a 21% increase in efficiency when processor and network performance are considered as compared to even partitioning. These results indicates the need to consider heterogeneity in processor and network performance, and heterogeneity in application computational complexity when partitioning mesh on distributed systems.

8 Acknowledgments

This work was supported by a NSF Young Investigator Award, contract number CCR-9357781, and a grant from AlliedSignal, Inc.

References

[1] Stephen T. Barnard and Horst D. Simon. A fast multilevel impelementation of recursive spectral bisection for partitioning unstrctured problems. Technical report, NAS systems Division, Applied Research Branch, NASA Ames Research Center, 1993.

[2] M. Berger and S. Bokhari. A partitioning strategy for non-uniform problems on multiprocessors. *IEEE Transactions on Computers*, C-36:5, 1987.

[3] H. C. Chen, H. Gao, and S. Sarma. WHAMS3D project progress report PR-2. Technical Report 1112, University of Illinois CSRD, 1991.

[4] Phyllis E. Crandall and Michael Quinn. A partitioning advisory system for network data-parallel processing. *Concurrency: Practice and Experience*, 7(5):479–495, August 1995.

[5] Charbel Farhat and Micel Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *International Journal for Numerical Methods in Engineering*, 36:745–764, 1993.

[6] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke. A wide-area implementation of the Message Passing Interface. *Parallel Computing*, 1998. to appear.

[7] I. Foster, J. Geisler, C. Kesselman, and S. Tuecke. Multimethod communication for high-performance networked computing systems. *Journal on Parallel and Distributed Computer*, to appear.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[9] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[10] Andrew S. Grimshaw, Wm. A. Wulf, and the Legion team. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1), January 1997.

[11] B. Hendrickson and R. Leland. The chaco user's guide. Technical Report SAND93-2339, Sandia National Laboratory, 1993.

[12] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical report, Sandia National Laboratories, June 1993.

[13] J. Jamison and R. Wilder. vbns: The internet fast lane for research and education. *IEEE Communications Magazine*, January 1997.

[14] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical report, Department of Computer Science,University of Minnesota, 1995.

[15] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 29:291–307, 1970.

[16] B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. K. Noor, editor, *Parallel Computations and Their Impact on Mechanics*. ASME, 1987.

[17] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, July 1990.

[18] Horst D. Simon. Partitiong of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135–148, 1991.

[19] Horst D. Simon and Charbel Farhat. Top/domdec: a software tool for mesh partitioning and parallel processing. Technical report, Report RNR-93-011, NASA, July 1993.

[20] Horst D. Simon, Andrew Sohn, and Rupak Biswas. Harp: A fast spectral partitioner. In *Proceedings of the Ninth ACM Symposium on Parallel Algorithms and Architectures*, Newport, Rhode Island, June 22-25 1997.

[21] Larry Smarr. Toward the 21st century. *Communications of the ACM*, 40(11):28–32, November 1997.

[22] Valerie E. Taylor and Jian Chen. A decomposition method for efficient use of distributed supercomputers for finite element applications. In *Proceedings of the 10th International Conference on Application-specific Systems, Architectures and Processors*, Chicago, Illinois, August 1996.

[23] Valerie E. Taylor and Jian Chen. Part: A partitioning tool for efficient use of distributed systems. In *Proceedings of the 11th International Conference on Application-specific Systems, Architectures and Processors*, Sweden, August 1997.

[24] D. Vanderstraeten, C. Farhat, P. S. Chen, R. Keunings, and O. Zone. A retrofit based methodology for the fast generation and optimization of large-scale mesh partitions: Beyond the minimum interface size criterion. Technical report, Center for Aerospace Structures, University of Colorado, September 1994.

[25] C. Walshaw, M. Cross, and M. Everett. Parallel dynamic graph partitioning for adaptive unstructured meshes. *J. Par. Dist. Comput.*, 47(2):102–108, 1997.

Appendix

Claim 1 *Partition problem for distributed systems is NP-complete.*

Proof 1 *We transform a proven NP-Complete problem, MINIMUM SUM OF SQUARES [8], to the Partition problem for distributed systems. Let set A , with $|A| = n$, and a size $s(a) \in Z^+$ for each $a \in A$ be an arbitrary instance of MINIMUM SUM OF SQUARES. We shall construct a graph $G = (V, E)$, with $|V| = n$, such that the desired partition exists for G if and only if A has a k, $k \leq n$, partition with minimum sum of squares.*

The basic units of MINIMUM SUM OF SQUARES instance are $a_i \in A$, $1 \leq i \leq n$. The local replacement substitute for each $a_i \in A$ is the collection E_i of 3 edges shown in Figure 5. Therefore, $G = (V, E)$ is defined as the following:

$$V = A \bigcup \bigcup_{i=1}^{n} \{a_i[j] : 1 \leq j \leq 2\}$$

$$E = \bigcup_{i=1}^{n} E_i$$

It is easy to see this instance of Partition problem for distributed systems can be constructed in polynomial time from the MINIMUM SUM OF SQUARES instance.

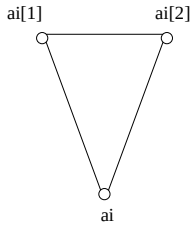

Figure 5. Local replacement for $a_i \in A$ for transforming MINIMUM SUM OF SQUARES to Partition problem for distributed systems.

If $A_1, A_2, ..., A_k$ are the disjoint k partitions of A such that the sum of squares is minimized, then the corresponding k disjoint partitions of V is given by taking $\{a_i[1], a_i[2], a_i\}$ for each a_i in every subset of A. We also restrict the cost function f_i to be the same as is in MINIMUM SUM OF SQUARES: $\left(\sum_{v \in V_i} s(v)\right)^2$ where $s(a_i[1]) = s(a_i[2]) = 0$ and therefore, $s(v) = s(a)$ for $v \in V_i$ and $a \in A_i$. This ensures that the partition $V_1, V_2, ..., V_k$ gives minimum sum of squares of the cost function.

Conversely, if $V_1, V_2, ..., V_k$ is a disjoint k partition of G with minimum sum of squares of the cost function, the corresponding disjoint k partition of set A is given by choosing those vertices a_i such that $\{a_i[1], a_i[2], a_i\} = v$ for some $v \in V_i$. Hence the minimum sum of squares for the cost function over k disjoint partitions ensures that the sum of squares of $s(a)$ on k disjoint set of A is also minimized. We conclude that the Partition problem for distributed systems is NP-Complete.